

Autonomy and software technology on NASA's Deep Space One

Richard Doyle
Jet Propulsion Lab
richard.j.doyle@jpl.nasa.gov

Douglas Bernard, Richard Doyle, Ed Riedel, Nicolas Rouquette, and Jay Wyatt, Jet Propulsion Laboratory
Mike Lowry and Pandurang Nayak, NASA Ames Research Center

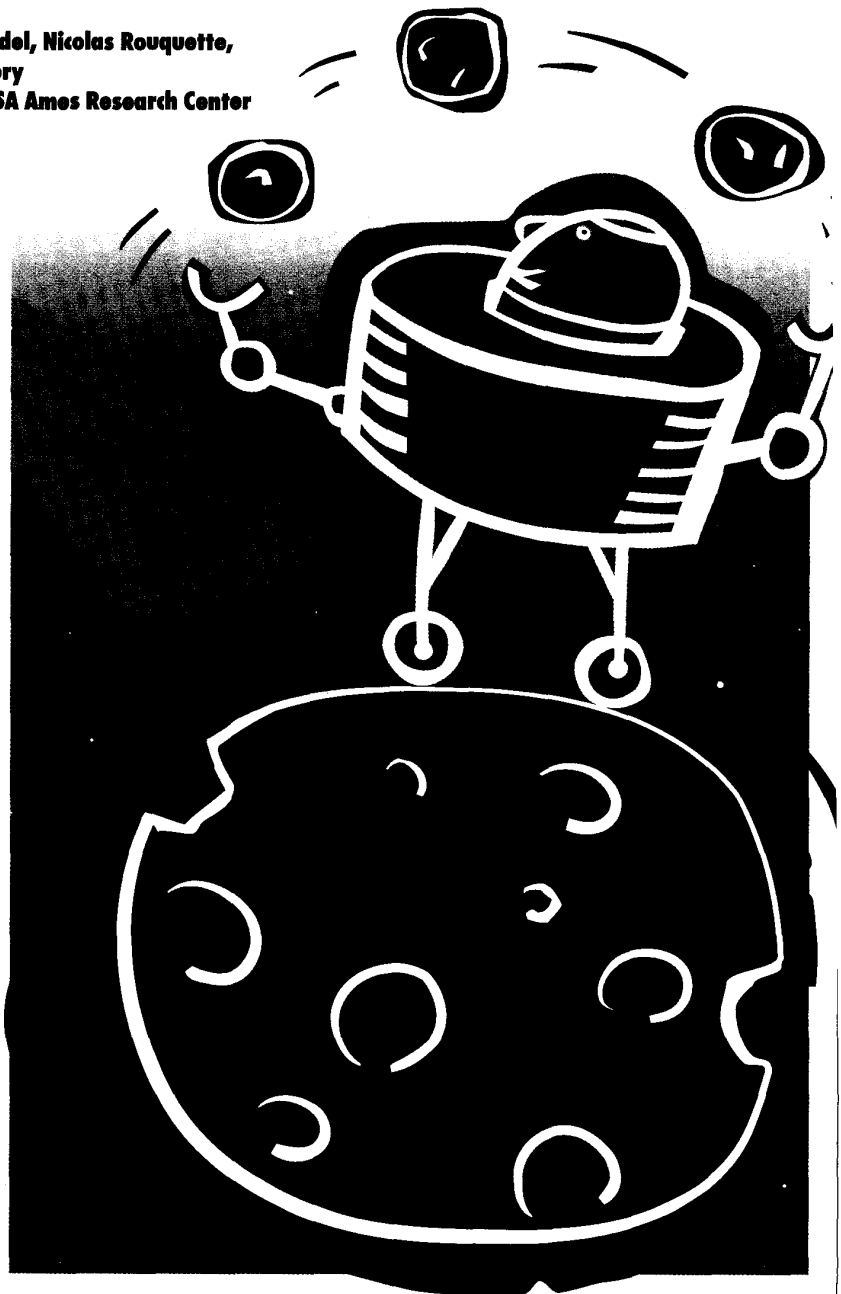
NASA's Deep Space One mission is unprecedented. Traditionally, NASA's space missions have been justified

by science data return as the primary, if not the sole consideration. DS1 is the first NASA mission whose main purpose is to demonstrate the flight readiness of a set of technologies. This first installment of the new "AI in Space" column will show how these AI-related technologies are helping to launch NASA into the exciting new era of autonomous space vehicles.

Deep Space One

DS1 is the vanguard of NASA's New Millennium Program, which was conceived to directly address the ongoing challenge of flight-qualifying technologies for mission use and to short-circuit the Catch-22 situation where flight project managers naturally prefer to utilize technologies only after they've been flown on another mission. Any of the DS1-qualified technologies might hold the key for enabling future NASA space-exploration missions. DS1 carries a dozen technology experiments, each demonstrating new capabilities that cover the gamut of spacecraft functions—from propulsion and sensing to power and communications. Three of these technology experiments demonstrate new capabilities in spacecraft autonomy and autonomous mission operations:

- The autonomy-related experiment with the largest scope on DS1 is a joint NASA Ames Research Center/Jet Propulsion Laboratory project called the *Remote Agent*.



AI in Space: The Era of Autonomous Space Systems

A new column launches this month, and follows the recent *IEEE Intelligent Systems* Special Issue on Autonomous Space Vehicles (Sept./Oct. 1998).

This column will report on results, progress, initiatives, and viewpoints relating to ongoing work at NASA and its partners to achieve autonomy capabilities in the space platforms that will support and accomplish the future NASA missions.

Topics will range from first examples of the use of autonomy capabilities in space, to new space exploration mission concepts conceived around autonomy, to challenges and concerns as seen by the NASA mission and scientist user communities, to status reports on autonomy technology development projects, to getting acquainted with the NASA family of AI and autonomy researchers and technologists.

This first installment of this column reports on technology flight experiments relating to autonomy on NASA's currently flying Deep Space One mission.

—Richard J. Doyle, Jet Propulsion Laboratory

Richard Doyle is the technical division manager of the Information Technologies and Software Systems Division and leader of the Center for Space Mission Information and Software Systems at JPL. He formerly held the roles of technical section manager of the Information and Computing Technologies Research Section and program manager for Autonomy Technology at JPL. He received his PhD in computer science at the MIT Artificial Intelligence Laboratory. He is a technical program chair for the Fifth International Symposium on Artificial Intelligence, Robotics and Automation for Space, at Noordwijk, The Netherlands, in June 1999. He gave the invited talk entitled "The Emergence of Spacecraft Autonomy" at the National Conference on Artificial Intelligence in Providence, Rhode Island, in July 1997. Contact him at richard.j.doyle@jpl.nasa.gov.

RA is both an autonomy architecture and a set of component reasoning engines for mission planning, execution, and fault protection.

- One of the fundamental space-mission functions is navigation. DS1 is the first interplanetary mission to be navigated by an *autonomous navigation* system. All previous missions have been navigated by ground operators. The DS1 navigation technology demonstration (AutoNav) enables a spacecraft to navigate independently of ground teams and ground links.
- As spacecraft begin to become more autonomous, mission operations concepts must also evolve. DS1's *Beacon Operations* experiment demonstrates a new end-to-end concept for mission operations, where the spacecraft takes responsibility for determining when ground support is needed.

Also, experience has shown that software-engineering issues emerge as autonomy capabilities are developed: one of the most daunting of these is testing. Encouragingly, a technique based in formal methods yielded important results on DS1 when an error was detected early in the development of an RA component.

Another software-engineering technique yielded crucial results on DS1 when our space technologists used automatic code generation to generate much of the core fault-protection flight code. Although not an official technology experiment, this success nonetheless enabled DS1 to meet its intense development schedule and launch in October 1998.

The Remote Agent

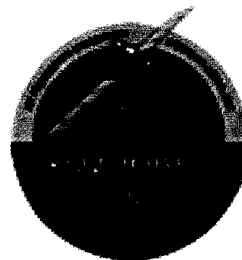
The Remote Agent experiment is a flight experiment that demonstrates a new approach to spacecraft command and control. In the

RA approach, the operational rules and constraints are encoded in the flight software.

The spacecraft operators therefore can consider the software as an autonomous remote agent in that they rely on the agent to achieve particular goals. The operators do not know the exact conditions on the spacecraft, so they do not tell the agent exactly what to do at each instant of time. They do, however, tell the agent exactly which goals to achieve in a period of time as well as how and when to report back. The DS1 mission will use this RA approach as an experiment.

The DS-1 RA experiment has several goals:

- provide an onboard demonstration of spacecraft autonomy, including goal-oriented commanding, time- and event-driven execution, and model-based fault diagnosis and recovery; and
- familiarize the spacecraft-engineering community with the RA approach and decrease the risk (both real and perceived) in deploying RAs on future missions.



The RA is formed by the integration of three separate AI technologies: an onboard planner-scheduler, a robust multithreaded executive, and a model-based fault-diagnosis and recovery system (see Figure 1). All three are written in Harlequin Lisp specifically ported to run under VxWorks on a RAD6000 processor. Each technology uses two distinct components: a general-purpose reasoning engine and application-specific models. The RA operates at several different levels of autonomy, ranging from traditional spacecraft commanding through onboard planning and execution.

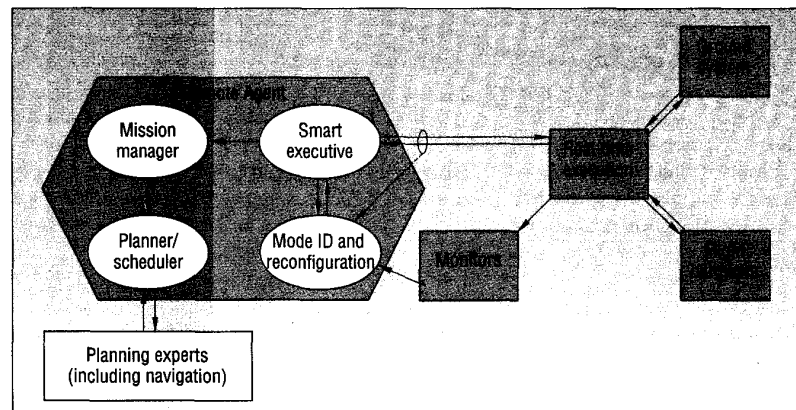


Figure 1. Remote Agent architecture.

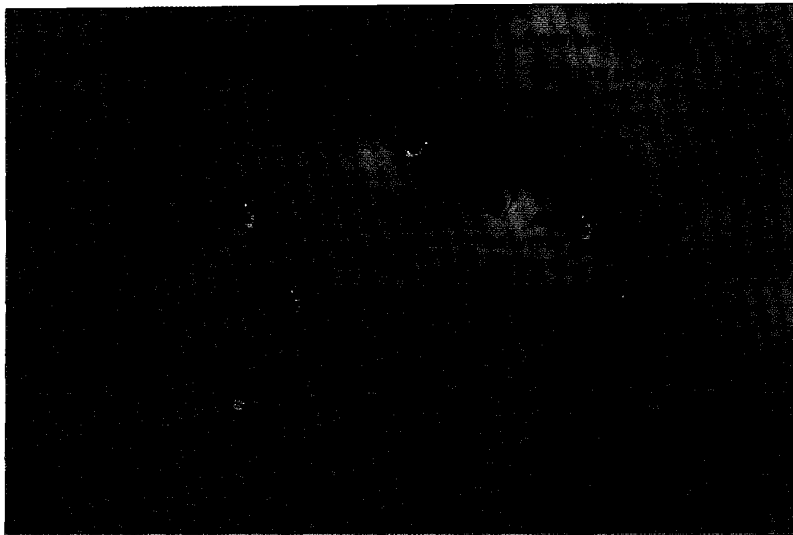


Figure 2. Comparing conventional and autonomous spacecraft navigation.

The RA preflight and flight validation proceeded as follows:

- First, its designers used the RA to handle low-level commands as instructed by the ground.
- Next, they demonstrated its ability to execute a flexible plan generated on the ground.
- Finally, they let the RA generate plans onboard and execute them without prior inspection of those plans by humans.

During the experiment, system designers injected several fictitious failures, letting the RA demonstrate its model-based fault-protection approach. With ground and flight testing now complete, all RA validation objectives have been met, and the team is applying the lessons learned during the experiment to future technology upgrades.

The RA team is led by JPL's Doug Bernard and ARC's Pandu Nayak.

The AutoNav system

By using images from an onboard camera of sufficient quality, the Autonomous Optical Navigation (AutoNav) system can control a spacecraft's flight path, including use of solar electric propulsion (SEP), and target one or more flyby encounters, or rendezvous (see Figure 2). Because the system was designed to be largely self-contained, it can be inserted into a fairly simple software architecture, without other autonomous systems required, except for attitude control. This is the situation for DS1. Even in the simple "traditional" environment of the borrowed Mars Pathfinder software set, AutoNav has achieved highly autonomous

behavior. This system's advantages and successful utilization so far have encouraged several missions to baseline its use in whole or part. These missions include the Space Technology-4 mission to rendezvous with and land on a comet, Stardust, which will use the close-approach system, and Deep-Impact, a Comet penetrator mission, which will use AutoNav's encounter and targeting components.

AutoNav consists of several subsystems and functions:

- *Navigation executive function.* The NavExec controls all AutoNav operations that cause physical action by the spacecraft. By communicating with the altitude control system (ACS), NavExec turns the spacecraft and images a series of target navigational beacons, which with DS1 are usually nearby asteroids. This complex of activities include planning the sequence of turns to optimize time utilization and insure completion of the photo-taking sequence on schedule. NavExec also performs similar duties during the long segments of SEP activity, during which NavExec commands the spacecraft to go to the required attitude, light the engine, and maintain thrust at periodically updated attitudes and magnitudes. Similarly, NavExec commands the execution of trajectory-correction maneuvers.
- *Image processing.* As its name implies, this function identifies the objects and stars in images relayed to AutoNav and does highly precise data reductions. Ultimately, we anticipate 0.1-pixel accuracy from the algorithms (although current scattered-light and other problems with an

onboard instrument have prevented this so far). Special-encounter image processing will amplify the dim signal of the target as seen many hours before closest approach.

- *Orbit determination (OD).* Using data from the image processor, AutoNav uses a batch-sequential modified Kalman filter to compute the spacecraft's position. It also estimates parameters modeling SEP thrust and random accelerations (such as errors in solar-pressure modeling, or spacecraft outgassing).
- *Maneuver planning.* With the results of the OD in hand, AutoNav will compute updates to the upcoming SEP thrust plan, or the components of a statistical trajectory-correction maneuver (that is, one based on statistical variations in the OD). These trajectory-correction maneuvers can use either SEP or the hydrazine propulsion system.
- *Encounter knowledge updates.* After the final trajectory-correction maneuver finishes, AutoNav switches to a special mode of activity, which specifically updates onboard knowledge of the target position and relays this information to the ACS for spacecraft pointing changes.

AutoNav began operations as soon as the spacecraft came to life after launch on October 24, 1998, providing critical ephemeris information (data on planetary positions) to the ACS. Over the following four months, mission developers checked out and invoked progressively more components of AutoNav, until April 20, 1999, when the spacecraft came completely under the control of the AutoNav system, flying a SEP-powered flight path computed onboard. This autonomous control will continue with periodic necessary suspensions or updates in AutoNav control for onboard tests and validations, leading to a fully autonomously controlled flyby of asteroid 1992KD on July 29, 1999.

JPL's Ed Reidel leads the Autonomous Navigation team.

Beacon Operations

This experiment aims to flight-validate an operations concept and the associated technology components necessary to enable more adaptive operations on future space missions. This approach will

- reduce the spacecraft-resource and mission-operation costs of the spacecraft-to-ground link,

- reduce the routine tracking burden of large-aperture antennas, which can help NASA's Deep Space Network ease the loading on its overconstrained antenna network, and
- reduce mission risk because the low-cost link can be maintained more frequently and also because a mission's telemetry link cannot be achieved due to spacecraft- or mission-design constraints.

Two subsystems implement the beacon operations functionality on DS1 (see Figure 3). The first is an end-to-end tone system that lets the spacecraft inform the ground whether data needs to be sent. This tone does not contain any telemetry, but rather represents one of four possible requests for ground action (no action required, contact when convenient, contact within a certain time, or contact immediately).

The second subsystem produces intelligent data summaries that are downlinked as telemetry after ground personnel respond to the tone request. Onboard summarization produces four types of engineering telemetry. This subsystem gathers high-level spacecraft information—such as the number of alarm crossings, spacecraft mode and state histories, and other pertinent statistics—since the last ground contact. It also gathers episode data for the culprit and causally related sensor channels whenever a sensor violates an alarm threshold and stores the data at a high sample rate. It collects snapshot telemetry at a much lower sample rate for all sensor and transform channels. Snapshot data serves only for rough correlation and to fill in the gaps between episodes. The last component of the downlinked summary, *performance data*, is similar to episode data but captures maneuvers or other events known in advance to be of interest to people on the ground. All of the summary algorithms are implemented in C for the VxWorks operating system.

The summary algorithms incorporate AI-based methods to enhance anomaly-detection and episode-identification capability. The ELMER (Envelope Learning and Monitoring using Error Relaxation) technology replaces traditional redlines with time-varying alarm thresholds to provide faster detection with fewer false alarms. The system uses a neural network to learn these functions; training can be performed onboard or on the ground (ground-based for DS1). ELMER is particularly powerful because it requires very little knowledge engineering and trains the neural net with nominal sensor data.

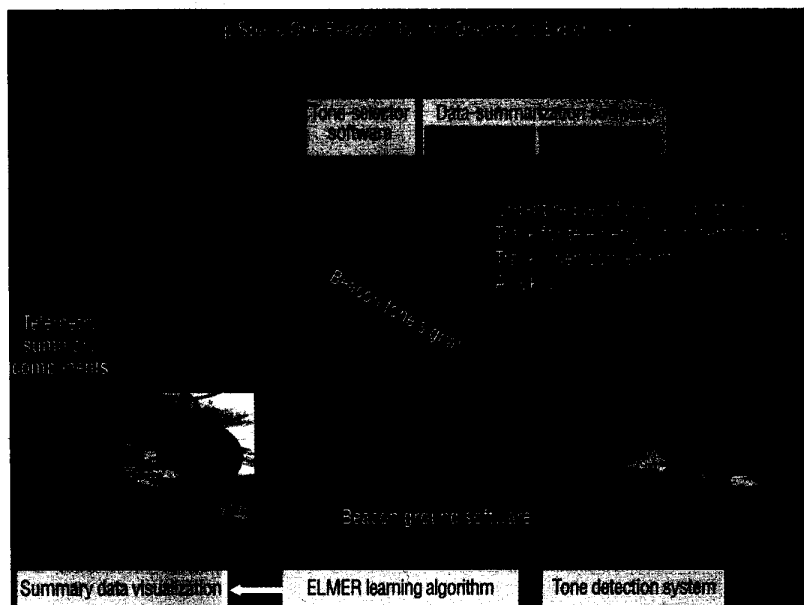


Figure 3. Beacon Operations system.

Another AI-based method produces empirical transforms that derive their heritage from previous AI research work at JPL in selective monitoring. Once computed onboard, these act as pseudosensors. The current transforms for DS1 compute high, low, and average values and first and second derivatives. Alarm limits can be placed on these transforms, and they can also serve as an input to the Elmer neural network. Additional transforms, if desired, can easily be defined and uplinked to the spacecraft as the mission progresses.

JPL's Jay Wyatt leads the Beacon Operations team.

Autonomy software testing

In space exploration, a major obstacle to widespread application of autonomy capabilities in flight software is not just technical feasibility; it is doubt about its trustworthiness as a replacement for human-in-the-loop decision-making. ACSs raise difficult verification and validation issues because, unlike conventional sequencer-based open-loop systems that perform transactions visible through uplink-downlink communications, they close many control loops and arbitrate many resources onboard with specialized reasoning in multiple concurrent threads. The number of possible execution paths for ACSs is many orders of magnitude greater than for traditional flight-control software. Verification and validation techniques that significantly increase confidence in these decision-making control systems are needed. Toward this end, researchers at NASA Ames, JPL, and Carnegie Mellon University demon-

strated techniques for verifying the greatly expanded number of possible execution paths inherent in autonomy software. Techniques were demonstrated for all parts of the Remote Agent—planner, executive, and mode identification and reconfiguration (MIR)—with the most extensive demonstration focusing on the resource manager of the goal-oriented executive.

Autonomy software is inherently concurrent; that is, multiple processes achieving different goals or subgoals execute in parallel. Concurrent-task software is easier to program than traditional sequences because the means of achieving each goal can be designed separately. Given autonomy's closed-loop nature, each goal being achieved represents a separate process. However, the extra degrees of freedom in achieving goals through separate processes can lead to unintended interactions between processes and lead to failures. These extra degrees of freedom make autonomy-software verification difficult through testing alone.

Previously, model-checking technology has been used to debug and verify concurrent digital hardware designs and communication protocols. Most of the demonstrations focused on using *model checking* to debug and verify portions of the RA. Model-checking is a set of mathematical algorithms based on automata theory for verifying and debugging concurrent or real-time systems modeled as interacting finite-state machines. Given a model and a property, a model-checker searches for traces of the model that violate the property—a trace is an interleaved sequence of states of the



Douglas Bernard is the technical group supervisor for the Deep Space 1 System Engineering group at JPL and team leader for the autonomy software autonomy technology development for the New Millennium mission. He received a BS in mechanical engineering and a master's degree from the University of Vermont, an MS in mechanical engineering from the University of California, and a PhD in aeronautics and astronautics from Stanford University. He has worked in dynamics analysis and attitude-control systems for spacecraft missions including the Galileo mission to Jupiter and the Cassini mission to Saturn. Contact him at douglas.e.bernard@jpl.nasa.gov.



Mike Lowry is a senior research scientist in the Systems Engineering Division of NASA Ames Research Center, and a member of the Automated Software Engineering group. His research interests include high-assurance program synthesis and verification of safety-critical systems for complex systems. He received his BS and MS in electrical engineering and computer science from MIT, and his PhD in computer science from Stanford University. Contact him at NASA Ames Research Center, Mail Stop 269-2, Moffett Field, CA 94035; lowry@pds.ames.nasa.gov.



Pandurang Nayak is a research scientist with the Systems Engineering Research Center, and a consulting faculty at Stanford University, where he received a PhD in computer science from Stanford. His research interests include model-based autonomous systems, diagnosis, and reasoning, both quantitative and qualitative and causal reasoning. He is an associate editor of *IEEE Transactions on Systems, Man, and Cybernetics*. Contact him at pnayak@ptolmty.arc.nasa.gov.



Ed Riedel joined the Voyager Mission Navigation team at JPL in 1977. He participated in the six planetary encounters of the Voyager mission and was the lead optical-navigation engineer for the Voyager 2 mission. He led the optical-navigation team for the Galileo mission to Jupiter and the Galileo tour, he became the New Millennium DS1 Navigation Team leader, directing the development of a completely new navigation system for the system to guide the approach and flyby of DS1 past the asteroid and comets. He is an author of numerous papers on navigation, orbital mechanics, and navigation-related image-processing algorithms. Contact him at joseph.e.riedel@jpl.nasa.gov.



Nicolas Rouquette is a senior computer scientist at JPL. He received a master's degree in electrical engineering from ESIHK in France and a PhD in computer science from the University of California, Berkeley. He is the software architect and engineer of the DS1 mission. His research interests are in AI, model-based reasoning, formal methods, and software engineering. His current research is in deploying statechart technology in embedded systems. Contact him at nicolas.f.rouquette@jpl.nasa.gov.



Jay Wyatt is the principal investigator for the Deep Space 1 Mission Experiment on DS1. He also manages the Instrumentation and Data Management work area at JPL, which funds research and development in mission operations, advanced telecommunications, and the management of the Deep Space Network. He is also the technical lead for the mission autonomy and systems-engineering research at the NASA Johnson Space Center & Computing Technology Research Section. He has worked in operations concepts for missions using autonomous systems in monitoring and diagnosis of complex systems. Contact him at wyatt@jpl.nasa.gov.

tem designers applied model checking directly to the core software. By exploring all possible execution states, model checking found five concurrency bugs. The errors found were for unusual situations that the designers had not fully considered. For example, model checking found a race condition when a task program was aborted and the locks it had on resources would not release correctly if the daemon monitoring the locks woke up at a particular point. For the planner and MIR, both of which are based on deductive rather than procedural methods, model checking was applied to validate the models. Specifically, model checking proved that it could find possibly unintended consequences of a model and thus help the model developer revise the model. Finally, NASA's designers found that runtime verification could be tied into the same framework as model checking through *behavior auditors* that monitor the runtime execution of autonomy software. Behavior auditors are specified in a language similar to the property descriptions used by model checkers.

ARC's Mike Lowry led the DS1 autonomy software testing work.

Auto-coding

Until DS1, JPL had not used code-generation techniques on a large scale for avionics software. However, the constraints of the mission design and development cycle and limited budget and resources dictated a departure from past practices. The demands of concurrent design and development as well as overlapping design and integration schedules would have drained all available resources allocated for system-level fault-protection. The requirement that postlaunch activities be directed by fault protection further increased the task's difficulty.

First, the DS1 project decided to reuse the successful Mars Pathfinder fault-protection (MPF) engine because it achieves a good separation of architecture and domain concerns. However, the MPF relied heavily on software engineers to encode the domain-specific fault-protection design into the target C language. Despite fewer software-engineering resources, DS1 faced a nonetheless larger design scope (spinner versus three-axis spacecraft) that also included postlaunch activities that, on other spacecraft, are typically handled with sequences. To meet this challenge, DS1's developers reorganized the fault-protection strategy to capture all designs in terms of high-level behavioral and

finite-state machines. Unlike simulators, model-checkers explore all relevant traces. In other words, they explore all realizable paths through the graph of states that can be reached from the initial state and that match the property being checked. Model-checkers are partic-

ularly well-suited to exploring the relevant execution paths of nondeterministic systems with multiple processes running in parallel. This makes them well adapted to verification and debugging of autonomy software.

For the procedural executive, NASA sys-

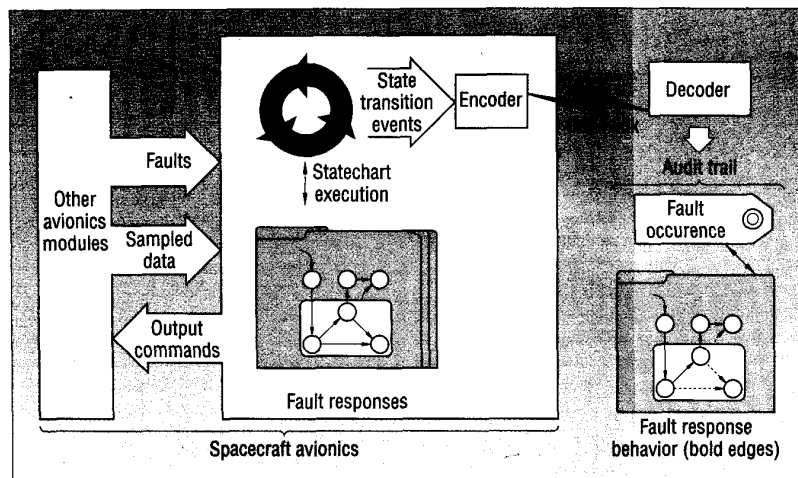


Figure 4. Behavior reconstruction in autocoding.

structural specifications instead of low-level C code. The Harel statechart notation became the standard means of describing the design of every fault-protection monitor and response. Such monitors are responsible for extracting features from raw data to reliably detect the occurrence of a known fault, while fault responses define the logic that controls the spacecraft to mitigate a fault's effects.

DS1 represents the first spacecraft at JPL where code generation from statecharts has been systematically applied to achieve rigorous and consistent software implementations in the target language directly from the statechart diagrams. This process was supported, in part, with the Mathwork's Matlab Stateflow toolbox. This toolbox provides a customizable translative code generator for statecharts. Extensive customizations of that toolbox were necessary—first, to address the needs of the DS1 fault-protection runtime architecture, and second, to fulfill the mission's end-to-end needs from design, to software, integration, and test, and to operations.

To support the systematic comparison of test results obtained from any pair of platforms (unit test, testbeds, and spacecraft), system developers had to adequately implement the execution of fault responses. They leveraged knowledge of the set of all faults and responses to derive a minimal-length encoding/decoding algorithm for onboard compression (encoding) and ground decompression (decoding) of state-transition events. Instead of instrumenting each fault response's state-transition code, the statechart-execution architecture signals state-transition events to the compression algorithm. From a sequence of such events, this algorithm computes an encoded value representing the path to the current state from the top-level statechart (the fault response handling the occur-

rence of a fault) through all intermediate statecharts invoked (that is, the hierarchy of helper subresponses invoked).

The encoded path and ancillary sampled variables constitute an event record. These algorithms produce the necessary and sufficient set of event records to provide full accountability of the rationale behind every state transition for every fault-response execution within memory limitations. This technique enables a behavior-reconstruction approach to testing—the process of producing a parsimonious explanation of a fault-response execution by determining the sequence of external events that recreates the same event record history as that obtained from the spacecraft (see Figure 4).

The autocoding work on DS1 was led by JPL's Nicolas Rouquette.

These five autonomy-technology experiments and related software-engineering activities on DS1 are paving the way for the use of autonomy capabilities in future NASA missions: proving technologies, reducing perceived risk, and ameliorating first-user costs. NASA is entering the era of autonomous space systems, and the results achieved on DS1 are already leading to applications of the autonomy technologies described here as well as inspiring additional autonomy-technology development work.

Acknowledgment

The research described here was performed at the Jet Propulsion Lab, California Institute of Technology, and at the Ames Research Center under contracts with the National Aeronautics and Space Administration. ■

III Intelligent SYSTEMS

Richard A. Anderson
Jet Propulsion Lab
Oscar F. Martinez
Stanford University
Yolanda Gil
University of Southern California
C. Lee Giles
NEC Research Institute
Kristian I. Hammar
University of Chicago
James V. Hansen
Brigham Young University
Marli A. Hearst
University of California, Berkeley
James A. Hendler
University of Maryland
Se June Hong
IBM T.J. Watson Research Center
Craig Knoblock
University of Southern California
Robert Laddaga
DARPA
Robert Plant
University of Miami
John R. Schaefer
University of California
John E. Swartout
University of Southern California
John T. Tjahjono
University of California
John W. Tomlin
University of California